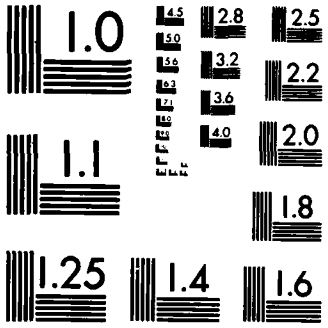

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

(2)

# 3-Year Research Progress Report for Grant No. AFOSR-82-0232 Programming Productivity Enhancement by the Use of Application Generators June 1, 1982 - May 31, 1985

by
Ellis Horowitz, Principal Investigator
Computer Science Dept.
University of Southern California
Los Angeles, California 90089
213-743-6453

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

AD-A158 590

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY<br>--- | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE<br>N/A | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S) | 5. MONITORING ORGANIZATION REPORT NUMBER(S)<br>AFOSR-TR- 85-0618 |
| 6a. NAME OF PERFORMING ORGANIZATION<br>University of Southern California — 6b. OFFICE SYMBOL (If applicable) | 7a. NAME OF MONITORING ORGANIZATION<br>AFOSR |
| 6c. ADDRESS (City, State and ZIP Code)<br>Los Angeles, CA 90089 | 7b. ADDRESS (City, State and ZIP Code)<br>Bldg. 410<br>Bolling AFB, D.C. 20332-6448 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>AFOSR — 8b. OFFICE SYMBOL (If applicable) NM | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>AFOSR-82-0232 |
| 8c. ADDRESS (City, State and ZIP Code)<br>Bldg. 410<br>Bolling AFB, D.C. 20332-6448 | 10. SOURCE OF FUNDING NOS. |

| PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT NO. |
|---|---|---|---|
| 61102F | 2304 | A2 | |

11. TITLE (Include Security Classification) Programming Productivity Enhancement by the Use of Application Generators

12. PERSONAL AUTHOR(S)
Ellis Horowitz

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Yr., Mo., Day) | 15. PAGE COUNT |
|---|---|---|---|
| Progress | FROM 10 Jan 85 TO 31 May 85 | 22 June 1985 | 14 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | |
|---|---|---|
| FIELD | GROUP | SUB. GR. |
| XX | XXXXXXXXX | XXX |
| | | |

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Application generators, Software engineering, Script Writer Software Development Environment

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The first chapter deals with the specific research areas that were investigated and discusses the accomplishments for each. The areas of research are: (i) Application Generators; (ii) Office Information Systems; (iii) Software Engineering; and (iv) the Script Writer Software Development Environment. The next chapter reviews the progress of all people who have been supported under the grant. Additional keywords: Ada programming language; computer applications

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☒ DTIC USERS ☐ | 21. ABSTRACT SECURITY CLASSIFICATION<br>Unclassified |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>John P. Thomas, Jr, USAF | 22b. TELEPHONE NUMBER (Include Area Code)<br>(202)767-5026 — 22c. OFFICE SYMBOL NM |

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

## Table of Contents

# List of Figures

# 1. Major Research Accomplishments

## 1.1 Introduction

This document summarizes the past three years of work under Air Force Office of Scientific Research Contract No. 82-0232. It is divided into several chapters. The first chapter deals with the specific research areas that were investigated and discusses the accomplishments for each. The areas of research are: (i) Application Generators, (ii) Office Information Systems, (iii) Software Engineering, and (iv) the ScriptWriter Software Development Environment. The next chapter reviews the progress of all people who have been supported under the grant. Two people have earned their Ph.D. degrees while performing work suppported by the grant. Several others are in various stages of completing their Ph.D. thesis, while others are just beginning graduate study. Chapter 3 lists all of the publications that have been written under grant support. Finally Chapter 4 discusses our plans for the next year.

## 1.2 Application Generators

When I originally wrote this research proposal, my focus was on the area of Application Generators. Systems such as RAMIS, NOMAD and FOCUS had all proven to be versatile at improving programmer productivity in the business sector. Their emphasis on nonprocedural programming and an interface to a database management system made them interesting candidates of study. My original intention was to examine the extent to which I could extend the Application Generator concepts to other worlds of programming. The first work to come out of this research was the paper [AppGenSur]. This paper made sense out of the various interpretations of nonprocedural programming and organized our understanding of their capabilities. We examined closely several systems and then defined generalized features based upon a generic model.

Our second activity was to see if we could design application generator features into a general purpose programming language. We decided to use Ada as the starting point. To begin we designed an extension of Ada that permits the language to interface naturally with a dbms. We made this extension a true language extension as opposed to simply a collection of procedure calls. We added a type **relation** and provided a broad set of operators including **select**, **project** and **join**.

```
type PERSON_REL is
       relation (key SS_NO)
        SS_NO : string(9);
        NAME : string(20);
        SEX : (F,M):
        SALARY : real;
end relation;

for P in PEOPLE where P.SEX = F loop
        PEOPLE[P].SALARY := PEOPLE[P].SALARY * 1.1
end loop;

PEOPLE := PEOPLE union NEW_GUY;
```

**Figure 1-1:** Example of AdaRel

In Figure 1-1 you see a small example that defines a relation, shows a for-loop that increases all people's salary in the relation by 10% and a final line that inserts a new person into the database. All of the features of AdaRel are fully defined in [AdaRel].

Our next step was to consider the report generation and graphic display features. This caused us to consider in general the question of designing applications that deal with screens of information. We observed that conventional languages are wholly inadequate as there input/output capabilities are built around the concepts of characters and lines. This led us to further extend the AdaRel model so that it includes a new basic unit called **screen**.

```
screen  <screen name>  ( <parameter list>) is
       format
              format definition
       end format;
       [activate
             <activation part>
       end activate;]
end screen;
```

**Figure 1-2:** Definition of Screen Type in AdaRel

In figure 1-2 you see the definition of an AdaRel Screen. Briefly, a screen has two parts: a format part and an activation part. The format part permits the definition of the screen, while the activate part performs a set of user determined actions at run-time. A most important aspect of the work was to show how data of type **relation** can be merged with the **screen** concept. Given these concepts, we were able to write many application programs and show how large volumes of data can easily be accomodated, while at the same time the user can interact with the screens choosing what he wishes to see. Complete definitions of the language extension plus examples was reported on in [HighLevelIO]. Finally, ideas about future directions of application generators were presented in [APPIDEAS].

## 1.3 Office Information Systems

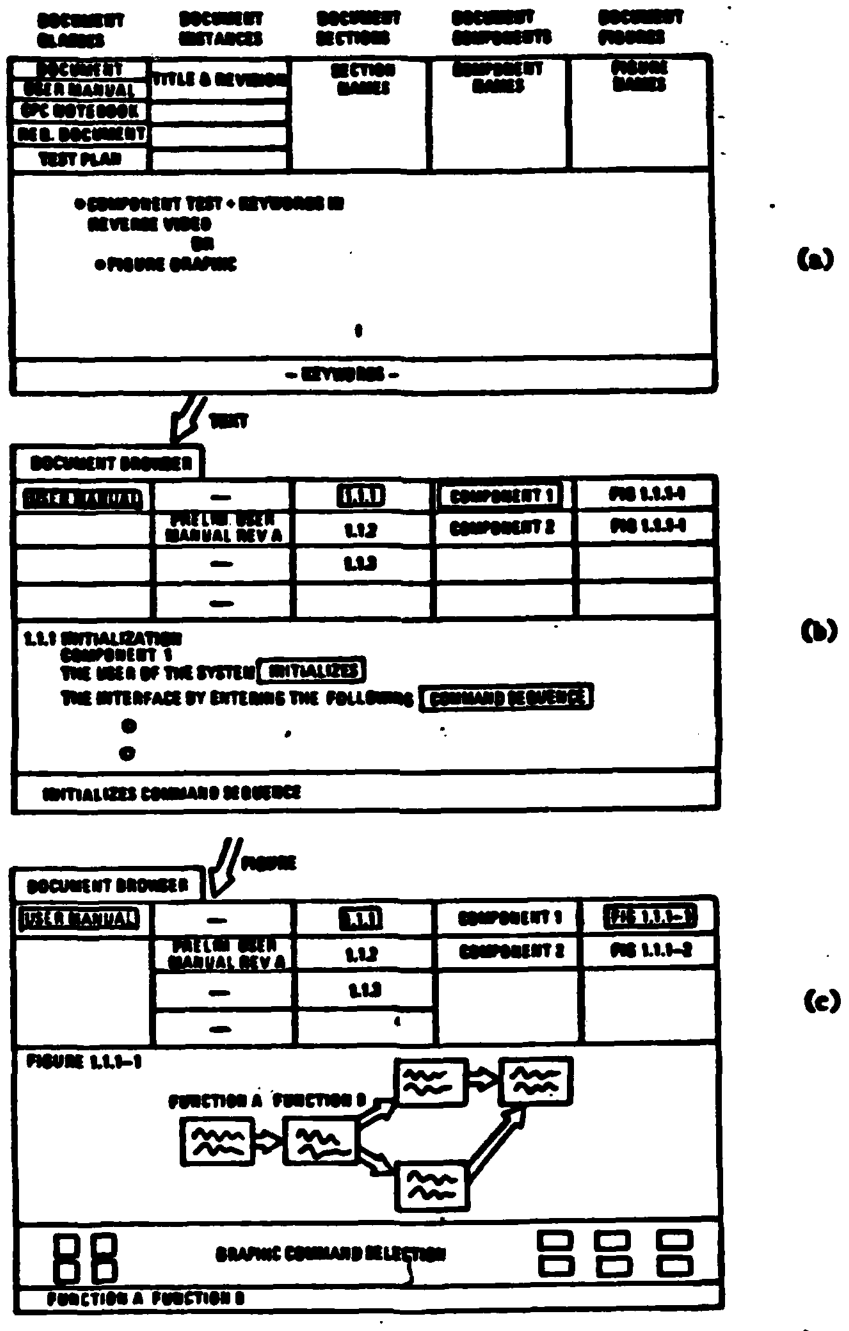
This work was undertaken between the principal investigator and Balaji Narasimhian. By studying the programming needs of offices, which are often data intensive, we hoped to be led towards new programming language facilities that support and enhance the database interface described previously. From this study we concluded that there was a major area of software development which is inadequately supported by current programming languages. This is the area of software that interacts with users, the user interface. Programming user interface applications is becoming a standard activity and yet, conventional programming languages have operators that deal with characters and lines and not with screens or sequences of screens. A second point of inspiration that resulted from the study is that the most common form of paper-user interaction is with a form. Therefore we concluded that a computer-based form seems a natual basis for an office information system. Pursuing such an environment, we have defined the basic properties of a form and the operations that must be supported by a forms-based system. These include: 1. form template definition; 2. form template instantiation; 3. specifying actions on form instances such as mailing, copying, saving and triggering; 4. validation of forms; and 5. storage and retrieval of forms and their contents. This work was summarized in the paper ***The Design of Office Information Systems***, [OIS].

## 1.4 Software Engineering

This area of study was done jointly with Ronald Williamson and the principal investigator. In its most general terms it is concerned with improving the overall productivity of the software development process. The approach is to try and capture all of the elements of this process as it is being produced and to place them in a database management system in a uniform way. This permits one to query the database of elements and to answer questions that concern these elements over the entire software life cycle.

Other researchers have taken the approach of defining a formal language into which requirements and design can be phrased. Our belief was that such formal methods, though attractive from the point of view of tool creation, expected too much from the programmer. Our work captures the life-cycle information *without* requiring knowledge of a special language. The developer enters his text as if it were an editor and only points to key elements. These are then automatically translated into data items for the dbms. By capturing the data in a way that requires little or no additional effort, we believe that our approach will be practical. A second aspect of our design was theoretical. We had to model the information in the database. This was done in terms of a graph model that has a special structure, namely a collection of trees with cross connections. Using the formal model, we then defined abstractly basic notions for retrieval of information across elements of the life-cycle.

Our design was followed by an actual prototype that was built and is running on a Xerox 1100 under the SMALLTALK environment. The use of the system is summarized in [SODOS:USE]. The definition of the programming concepts and the use of an object-based methodology is given in [SODOS:Definition]. The actual implementation is discussed in [SODOS:Implementation]. Figure 1-3 shows the SODOS document browser window in three stages. In the first stage one sees 5 panes at the top. Each pane is instantiated as the document is further refined. In the second stage one sees the User Manual has been selected and section 1.1.1 is being defined. In stage three a figure is being entered into the document and into the database.



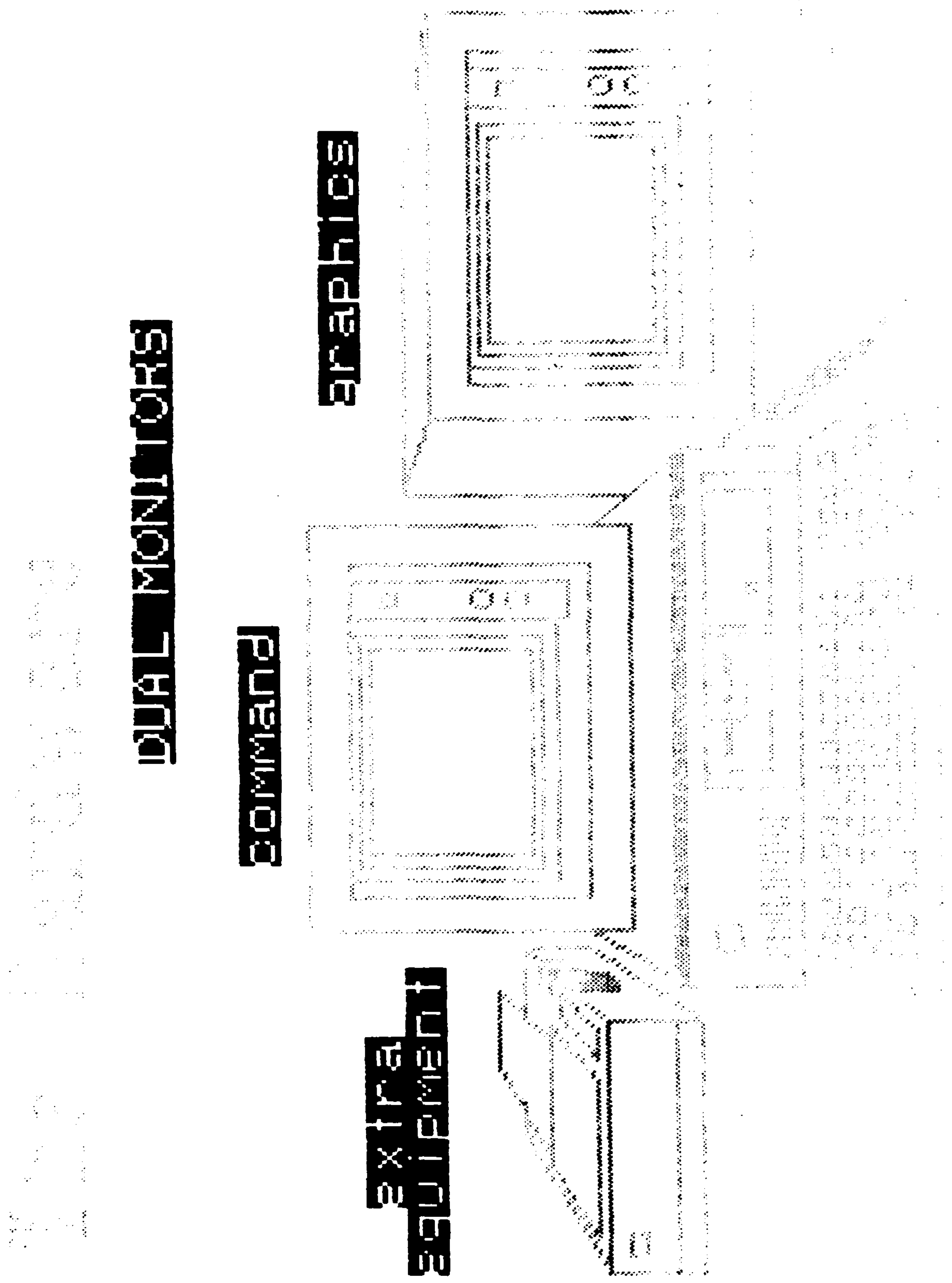
The SODOS Document Browser Window

## 1.5 SCriptWriter

The path of true love and research is often not straight. To understand the SCriptWriter Software Development Environment, it is useful to consider how we arrived at such a project. We were led to *SCriptwriter* from two directions: one being our research on Application Generators and the other being the particular computer systems we purchased. From the Application Generator work we realized that better systems could only be built if the domain of application was well focused. A second conclusion was that there was a great need for tools to design systems that involve interactive screens of information. Our second influence was the work being done on our IBM PC/ATs and the observation that they represented a delivery system upon which computer-based instruction could be run for large numbers of people. Thus we set ourselves the goal of devising a software development environment that would be suitable for the task of producing interactive, screen-based applications on microcomputers.

*SCriptWriter* is a software development environment that supports the development of multi-media productions. Its hardware consists of a dual monitor IBM PC/AT attached to a digitizer, sound chip and laser disk. An illustration can be seen in Figure SC. The software consists of 4 basic components: a command interface, a disk manager, an object-based programming language and a collection of editors. The editors are available for handling text, graphics, animation, laser disk, and font definition.

Though I cannot discuss all of the features encompassed in the system in this summary, I will point to two main features. One is the metaphor that is used. Creating a production can be quite complicated. The author needs some mental model so he can be guided as he creates his production. The metaphor employed in *ScriptWriter* is that of a play. Just as a play has actors, scenery, director and stagehands, so does *ScriptWriter*. A second major feature is the *IQ* programming language. This is an object-based programming language that includes the notion of player and lines as high-level concepts in the language. One goal we have attempted to achieve is to make the environment commands consistent with the language. By this I mean that all operations


DUAL MONITORS
graphics
command
extra
equipment

that can be performed at the environment level can also be performed in the programming language.

Over the past year of the grant we have designed the *SCriptWriter* system and begun its development. It is the activity that has dominated our time and collectively we are all quite excited about the research. The current design of the system is described in [SC REF]. A study of the design of the user interface is discussed in [SC UI].

## 2. Staff

The following people have been supported on this grant during its duration.

| | |
|---|---|
| Horowitz, Ellis | Principal investigator |
| Kemper, Alfons | Graduated 8-1-84 with his Ph.D. Currently Research Assistant Professor at Univeristy of Karlsruhe |
| Williamson, Ronald | Graduated 10-1-84 with his Ph.D. Now Research Scientist at Hughes Aircraft Corp. |
| Narasimhan, Balaji | Currently a graduate student in Computer Science |
| Papa, Marco | Currently a graduate student in Computer Science. working on his Ph.D. thesis. Working title is "Programming Abstractions for Computer Animation" |
| Bills, Mark | currently working on the development of the IQ language, a part of SCriptWriter, while working on a B.S. degree in Computer Science. Entering the M.S. program in Fall '85. |
| Anderson, David | Currently working on the development of the IQ language, a part of SCriptWriter, while working on a B.S. degree in Computer Science. |
| Garg, Pankaj | Graduate student in Computer Science just starting to work towards his Ph.D. degree |

## 3. Publications

- [AppGenSur] "A Survey of Application Generators", *IEEE Software*, vol. 2 No. 1, Jan. 1985, 40-54 (with Kemper and Narasimhan)

- [AdaRel] "AdaRel: A Relational Extension of Ada", Computer Science Technical Report U.S.C., October 1983.

- [HighLevelIO] "High-Level Input/Output Facilities in a Database Programming Language", *Proc. International Conf. on System Sciences*, Jan. 1985, 67-80. (with A. Kemper)

- [APPIDEAS] "Application Generators: Ideas for Programming Language Extensions", *Proc. ACM Annual Conf.*, San Francisco, Oct. 8-10, 1984, 94 - 101. (with A. Kemper and B. Narasimhan).

- [OIS] "The Design of Office Information Systems", Technical report TR-83-320, Computer Science Dept., U.S.C. December, 1983.

- [SODOS:Use] "SODOS A Software Documentation Support Environment: Its Use", Computer Science Dept., U.S.C. Sept. 1984 submitted to *IEEE Trans. on Software Engineering*

- [SODOS:Definition] "SODOS A Software Documentation Support Environment: Its Definition", Computer Science Dept., U.S.C. Sept. 1984 submitted to *IEEE Trans. on Software Engineering*

- [SODOS: Implementation] "SODOS A Software Documentation Support Environment: Its Implementation", Computer Science Dept., U.S.C. Sept. 1984 submitted to *IEEE Trans. on Software Engineering*

- [SODOS:Summary] "SODOS A Software Documentation Support Environment: Summary", *Proc. IEEE Software Engineering Conference*, London, Aug. 1985 to appear.

- [SC REF] "Scriptwriter Reference Manual" Computer Science Dept., U.S.C. April, 1985.

- [SC-UI] "An Analysis of the SCriptWriter User Interface" Computer Science Dept., U.S.C. May, 1985.

- [SC PROP] "SCriptWriter Software Development Environment", Proposal submitted to AFOSR covering the period July '85 - June '88.

## 4. Plans for the Future

The continuing development of *SCriptWriter* is leading us to many interesting issues in computer science. One issue is the design of an object-based programming language that effectively deals with animation definition. Other object-based languages such as SMALLTALK, though powerful and general, consequently require a great deal of computer resources. This fact makes them virtually unusable on microcomputers and moreover their ability to do animation is highly restricted. Our language, *IQ* presents a compromise between capability and speed. In the next year we plan to develop a full set of animation definition capabilities. This will consist both of an environment editor and a analagous set of language features.

Another important development that is planned for the coming year is the development of a user interface for interactive laser disk presentations. IBM is supplying us with an IBM PC XT configured to a laser disk. We plan to port the existing *SCriptWriter* system to the configuration and then build the laser disk editor afterwards.

Another plan that we have for *ScriptWriter* in the coming year is to develop notation for talking about structured graphical images. For example, if a production involves a clown, then one wants to be able to refer to the clown, or to hit hat, arms and legs. Each of these elements can have actions associated with them. We are working on language notation that supports the definition of heirarchically related graphical objects and their manipulation.

Finally a complete discussion of *SCriptWriter* and related work is contained in [SC PROP].